LABORATÓRIO DE ALTA EXATIDÃO E ALTO DESEMPENHO

Carlos Amaral Hölbig[1]
Rafael Linden Sagula[2]
Úrsula A L Fernandes[3]
Tiarajú Asmuz Diverio[4]
Dalcidio Moraes Claudio[5]

Instituto de Informática e CPGCC da UFRGS
Caixa Postal 15064 - 91501-970 Porto Alegre - Brasil
E-mail: {holbig, sagula, ursula, diverio, dalcidio}@inf.ufrgs.br

RESUMO

Este trabalho descreve o sistema de software *Laboratório de Alta Exatidão e Alto Desempenho*, desenvolvido para resolução de problemas da computação científica com verificação automática do resultado. O sistema é composto por um núcleo de alta exatidão, pela biblioteca de rotinas básicas intervalares (*libavi.a*) e pelos módulos aplicativos intervalares. Entre as características principais desse sistema estão a verificação automática do resultado pelo computador, a alta exatidão e o alto desempenho que são decorrentes do uso do supercomputador Cray Y-MP e da linguagem Fortran 90.

ABSTRACT

This paper describes the system of *High Accuracy and High Performance Laboratory* software developed for solution of problems of the scientific computing with automatic result verification. The system is composed of a high accuracy kernel, the library of interval basic routines (*libavi.a*) and interval applied modules. Among the main features of this system are the automatic result verification by the computer, the high accuracy and the high performance, which results from the use of Cray Y-MP supercomputer and Fortran 90 language.

---

[1] Meste em Ciência da Computação (CPGCC/UFRGS, 1996)
[2] Acadêmico em Ciência da Computação (UFRGS)
[3] Mestranda em Ciência da Computação (CPGCC/UFRGS)
[4] Professor Dr. CPGCC/UFRGS, Coordenador Departamento de Informática Teórica (II/UFRGS)
[5] Professor Dr. CPGCC/UFRGS, Líder do Grupo de Matemática da Computação (CPGCC/UFRGS)

1 INTRODUÇÃO

Este artigo relata o sistema de *software* denominado de Laboratório de Alta Exatidão e Alto Desempenho, desenvolvido para o ambiente do supercomputador Cray Y-MP pelo Grupo de Matemática da Computação da UFRGS. A finalidade desse sistema é proporcionar um ambiente onde problemas da computação científica possam ser resolvidos com rapidez, exatidão e confiabilidade, e onde a verificação do resultado seja feita automaticamente pelo computador. Para tanto, foram incorporados nesse sistema alguns conceitos da Matemática da Computação e de Aritmética Computacional, como: aritmética de alta exatidão (definida por Kulisch em [KUL81]), matemática intervalar, produto escalar ótimo, verificação automática do resultado e métodos de inclusão baseados no Teorema de ponto-fixo de Brouwer, os quais são revisados nos próximos itens.

O Laboratório é composto por um núcleo de aritmética de alta exatidão, uma biblioteca intervalar básica e módulos aplicativos intervalares, como por exemplo resolução intervalar de sistemas de equações lineares e integração numérica com verificação automática.

1.1 Aritmética de Alta Exatidão

A aritmética de alta exatidão possibilita que os cálculos sejam efetuados com máxima exatidão. Para isto, é necessário que o formato ou tipo de dado e as operações aritméticas suportadas pelo *hardware* ou pela linguagem de programação satisfaçam as condições de um semimorfismo ([KUL81]). Todas as operações construídas em cima desta definição proporcionam resultados com máxima exatidão. Isto é, o resultado difere do valor exato no máximo em um arredondamento.

O Padrão de aritmética IEEE surgiu com o objetivo de padronizar a aritmética de ponto-flutuante em todas as plataformas. Este foi um passo inicial para a obtenção da aritmética de alta exatidão. Esse Padrão não especificou os arredondamentos para variáveis complexas, operações entre vetores e matrizes e, também, não incluiu o produto escalar ótimo, essencial para a garantia da alta exatidão nos cálculos. Por causa disso, a GAMM/IMACS propôs outro padrão (ver [IMA91]). Os requisitos da aritmética de alta exatidão são: arredondamentos direcionados e operações aritméticas com máxima exatidão. Arredondamentos direcionados são funções de mapeamento (ver [KUL81]) utilizadas para representar os números reais em números de máquina. Quando a função de mapeamento é aplicada a qualquer elemento do conjunto dos números de máquina ela produz o próprio número de máquina. Para se ter implementada uma aritmética de alta exatidão, deve-se ter os dois tipos de arredondamentos direcionados:

- Arredondamento para cima (Δx) é a função que aproxima o número real x para o maior número de máquina que o contém.
- Arredondamento para baixo (∇x) é a função que aproxima o número real x para o menor número de máquina que o contém.

Esses arredondamentos são necessários para se ter a garantia nos cálculos, sendo fundamentais para a implementação da aritmética intervalar de máquina. Outro tipo de arredondamento é o arredondamento simétrico (ou para o número mais próximo de máquina). Este arredonda o número real x para o número de máquina mais próximo. É definido em função dos arredondamentos anteriores e produz um menor erro de aproximação.

Operações aritméticas com máxima exatidão são necessárias para se ter uma aritmética de alta exatidão. Elas são definidas de forma que só um arredondamento é aplicado nas operações aritméticas básicas, resultando que o valor calculado e o valor exato diferem por apenas um arredondamento. Se $\circ$ é uma operação aritmética no espaço dos números reais R, então a correspondente operação no computador $\boxplus$ no conjunto dos números de máquina F é definida por: $x \boxplus y := \square(x \circ y)$ para todos $x,y \in F$. (onde $\square$ é um arredondamento). Isto é, a operação no computador deve ser efetuada como se o resultado exato fosse calculado primeiramente e, então, aproximado com o arredondamento selecionado.

Resumindo, operações semimórficas para números reais e complexos, vetores e matrizes, assim como para intervalos reais e complexos, vetores e matrizes intervalares, podem ser realizadas em termos de 15 operações aritméticas fundamentais em aritmética de ponto-flutuante: +, -, *, /, ., cada uma com os arredondamentos para cima, para baixo e para o mais próximo.

1.2 Aritmética Intervalar de Máquina

A aritmética intervalar trata com dados na forma de intervalos numéricos e surgiu com o objetivo de automatizar a análise do erro computacional, trazendo uma nova abordagem que permite um controle de erros com limites confiáveis, além de provas da existência ou não da solução de diversas equações.

Ao invés de serem aproximados para números de ponto-flutuante mais próximos (por alguma das regras de aproximação ou dos tipos de arredondamento), os reais são representados por intervalos de números em ponto-flutuante, ou seja, um real x é representado por um intervalo X=[x1, x2], onde os limites inferior (x1) e superior (x2) são números de máquina, tais que $x1 \leq x \leq x2$. Todas as operações aritméticas, operadores relacionais, funções elementares, são definidas para argumentos intervalares. A partir da aritmética intervalar são desenvolvidos os conceitos que compõem a matemática intervalar e os métodos intervalares para resolução de problemas numéricos. A figura abaixo mostra algumas das operações existentes sobre o conjunto de intervalos de ponto-flutuante.

Inverso aditivo:	$-X = [-x1, -x2]$
Pseudo inverso multiplicativo:	$1/X = [1/x2, 1/x1] \quad 0 \notin X$
Adição:	$X+Y = [x1 \nabla^{+} y1, \; x2 \; \Delta^{+} y2]$
Subtração	$X-Y = [x1 \nabla^{-} y2, x2 \; \Delta^{-} y1]$
Multiplicação:	$X*Y = [\min\{ x1 \nabla^{*} y1, x1 \nabla^{*} y2, x2 \nabla^{*} y1, x2 \nabla^{*} y2\}, \max\{x1 \Delta^{*} y1, x1 \Delta^{*} y2, x2 \Delta^{*} y2, x2 \; \Delta^{*} y2\}]$
Divisão:	$X/Y = [\min\{ x1 \nabla^{/} y1, x1 \nabla^{/} y2, x2 \nabla^{/} y1, x2 \nabla^{/} y2\}, \max\{x1 \Delta^{/} y1, x1 \Delta^{/} y2, x2 \Delta^{/} y2, x2 \Delta^{/} y2\}]$
Intersecção:	$X \cap Y = \{ [\max\{x1, y1\}, \min\{x2, y2\}]$, se $x1 \leq y2$ e $y1 \leq x2$, $\emptyset$ - caso contrário.
União:	$X \cup Y = \{ [\min\{x1, y1\}, \max\{x2, y2\}]$, se $X \cap Y \neq \emptyset$, *ERRO* - caso contrário.
Distância:	$d(X,Y) = \max\{ \lvert x1-y1 \rvert, \lvert x2-y2 \rvert\}$
Valor absoluto:	$\lvert X \rvert = d(X,[0,0]) = \max\{ \lvert x1 \rvert, \lvert x2 \rvert\}$
Diâmetro:	$D(X) = x2-x1$

Figura 1 - Operações Básicas Intervalares

1.3 Princípios da Verificação Automática do Resultado

A Verificação Automática do Resultado tem como objetivos computacionais o controle sobre: o erro de arredondamento, o erro de aproximação e o trabalho com dados de entrada incertos (intervalos).

O controle do erro de arredondamento é conseguido por meio da aritmética intervalar. Com isso, a aritmética intervalar permite o cálculo de extremos seguros para a solução de um problema numérico.

Na adaptação do computador para o controle automático de erros, a sua aritmética teve de ser estendida ainda a um outro elemento. Todas as operações com números de ponto-flutuante (adição, subtração, divisão, multiplicação e produto escalar ótimo de vetores em ponto-flutuante) devem ser supridas com os arredondamentos direcionados.

A magnitude do erro de aproximação é diretamente proporcional ao diâmetro do intervalo resultante, isto é, quanto maior for o diâmetro do intervalo maior será o erro de aproximação. Às vezes é possível diminuir o diâmetro do intervalo resultante usando os métodos de inclusão.

A Verificação Automática do Resultado é baseada em três pré-requisitos: na Aritmética Intervalar, no Produto Escalar Ótimo e em Algoritmos Apropriados. No cálculo tradicional do produto escalar de dois vetores com n componentes cada, envolve $2n$-1 arredondamentos. Se ocorrerem cancelamentos catastróficos, um grande número de dígitos significativos podem ser perdidos. O resultado final poderá estar totalmente errado. A forma ótima do cálculo do produto escalar se dá através de um resultado intermediário, calculado com precisão infinita e com faixa de expoente ilimitada e, então, esse resultado é arredondado para o formato em ponto-flutuante desejado de acordo com o arredondamento selecionado.

1.4 Métodos de Inclusão e Teorema do Ponto-fixo de Brouwer

Muitos algoritmos de verificação numérica são baseados nos teoremas de ponto-fixo com respeito a conjuntos intervalares. O teorema mais conhecido e utilizado é o de Brouwer, que será citado a seguir (ver [HAM93]).

Teorema de Ponto-fixo de Brouwer: Deixando $f: R^n \to R^n$ ser um mapeamento contínuo e $X \subseteq R^n$ ser um conjunto fechado, convexo e limitado. Se $f(X) \subseteq X$, então f tem no mínimo um ponto-fixo[6] x^* em X.

Seja $X \in IR^n$ um vetor intervalar de máquina. Como uma caixa no espaço n-dimensional, X satisfaz as condições do teorema de ponto-fixo de Brouwer. Suponha-se que se pode encontrar uma caixa com $f(X) \subseteq X$. Então, X contém pelo menos um ponto-fixo x^* de f. As suposições anteriores valem se f for substituído pela avaliação intervalar de ponto-fixo $f_\Diamond$, pois $f_\Diamond(X) \subseteq X$ implica $f(X) \subseteq X$, desde que $f_\Diamond(X)$ é um superconjunto[7] de $f(X)$.

Teorema de Ponto-fixo de Brouwer Modificado: Sendo $f: R^n \to R^n$ um mapeamento contínuo e $X \subseteq R^n$ ser um conjunto fechado, convexo e limitado. Se $f(X) \overset{\circ}{\subset} X$, então f tem um único ponto-fixo x^* em X. A relação estar contido no interior (o intervalo X está contido no interior de Y) é definida por $X \overset{\circ}{\subset} Y \Leftrightarrow \underline{y} < \underline{x}$ e $\overline{x} < \overline{y}$. Nota-se que esses dois teoremas são

---

[6] x é dito um ponto fixo de f, quando $x=f(x)$.

[7] x é dito um superconjunto de y se x contém y, ou seja, $x \supseteq y$,

muito parecidos e de fácil aplicação em algoritmos, porque só trabalham com os extremos do intervalo.

Uma das aplicações do **Teorema de Ponto-fixo de Brouwer** é no projeto de algoritmos com verificação dos resultados. Primeiro, deve-se achar uma fórmula de iteração de ponto-fixo $f(x) = x$ equivalente ao problema original e, toma-se $f_\Diamond$ como uma extensão de máquina da fórmula de iteração intervalar de um método numérico (Newton, Gauss-Seidel, etc.). Inicializa-se o algoritmo com alguma aproximação da solução $X^{(0)}$, então

$$X^{(k+1)}=f_\Diamond(X^{(k)}), \text{ para } k= 0, 1, 2,...$$

Se para alguma iteração, com $k \geq 0$, for encontrado $X^{(k+1)} \subseteq X^{(k)}$, então é provado[8] que o problema original tem no mínimo uma solução x^* contida em $X^{(k)}$. Unindo-se a aritmética intervalar e os dois teoremas, tem-se os Métodos de Inclusão. Com o objetivo de garantir a existência e/ou a unicidade da solução de um algoritmo, pode-se distinguir dois tipos de métodos de inclusão: *a priori* e *a posteriori*, com relação a aproximação inicial.

Para o **Método *a priori***, a aproximação inicial já inclui o ponto-fixo procurado. Neste caso, a fórmula de iteração (1.1) deve ser modificada por uma fórmula de interseções sucessivas

$$X^{(k+1)}=f_\Diamond(X^{(k)}) \cap X^{(k)}, \text{ para } k= 0, 1, 2,...$$

onde $X^{(k)} \in IR^n$ um vetor intervalar de máquina na k-ésima iteração e $f_\Diamond$ uma extensão de máquina da fórmula de iteração intervalar.

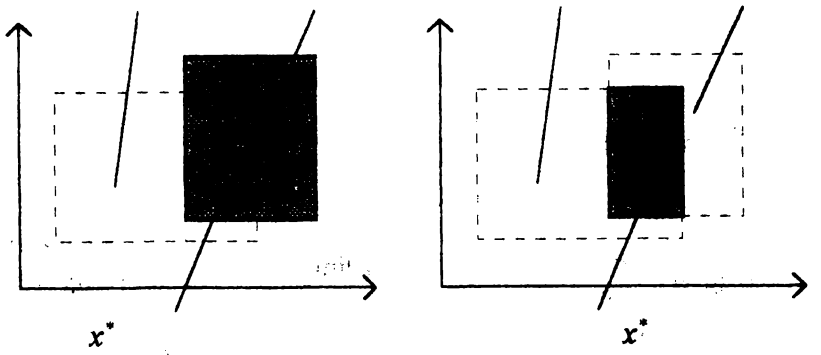

Figura 2 - Método de Inclusão *a priori*.

No **Método *a posteriori***, o ponto-fixo procurado não está, necessariamente, contido na aproximação inicial. Aqui, espera-se que as iterações se aproximem cada vez mais do ponto-fixo, até incluí-lo. Quanto melhor for a aproximação inicial (mais próxima do ponto-fixo), mais rápida será a convergência do algoritmo. Entretanto, na prática, as iterações vão se aproximando do ponto-fixo x^*, mas muitas vezes não conseguem incluí-lo. Um truque simples é utilizado para resolver esta limitação. Antes de se iniciar uma nova iteração, a mesma iteração é aumentada em seus extremos por uma inflação épsilon (ε).

[8] Provado por Sigmund Rump em [KUL83].

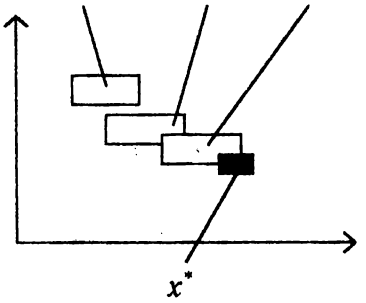

Figura 3 - Método de Inclusão *a posteriori*.

Métodos que usam os teoremas de ponto-fixo podem ser modificados para provar a unicidade de um ponto-fixo, com o teorema de Brouwer modificado. Para tanto, usa-se a fórmula $X^{(k+1)} \overset{\circ}{\subseteq} X^{(k)}$, para $k = 0, 1, 2,...$, como critério de parada.

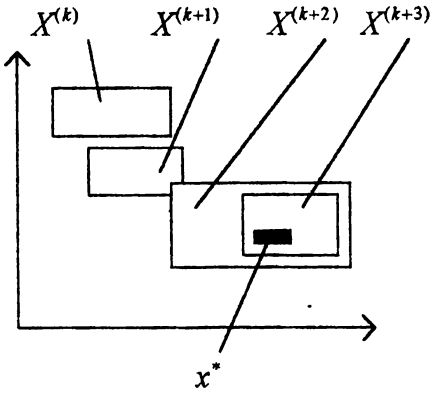

Figura 4 - Critério de parada para provar a unicidade de uma solução.

Na iteração $X^{(k+3)}$ existe apenas uma única solução, pois, $X^{(k+3)} \overset{\circ}{\subseteq} X^{(k+2)}$ (critério de parada).

2 AMBIENTE DE DESENVOLVIMENTO

O ambiente de desenvolvimento do Laboratório de Alta Exatidão e de Alto Desempenho foi o Supercomputador Cray com a linguagem FORTRAN 90, que se encontra no Centro Nacional de Supercomputação (CESUP/RS). Ele é da família Y-MP, com dois processadores vetoriais contendo as seguintes características: velocidade total máxima de 660 *MFlops*; palavra de 64 *bits*; memória RAM de 256 *Mbytes* e 16 *Gbytes* de disco e sistema operacional UNICOS, compatível com *UNIX System* V (ver [CRA93]).

O processamento escalar ocorre seqüencialmente e usa um operando ou um par de operandos para produzir um único resultado. O processamento escalar é efetuado usando registradores escalares. A principal vantagem do processamento vetorial sobre o escalar é a eliminação do tempo de carga (*startup*) de todas as operações. O tempo de carga para operações vetoriais é suficientemente pequeno para que o processamento vetorial seja mais eficiente do que o processamento escalar para vetores contendo poucos elementos.

No Cray do CESUP/RS, estão disponíveis os compiladores FORTRAN (77 e 90), C e Pascal com subrotinas científicas e matemáticas altamente otimizadas, incluindo Boeing CS, BLAS3, LINPACK, EISPACK, FFT e operações matriciais.

O FORTRAN ainda é a principal linguagem de programação utilizada na área científica. O mesmo vem evoluindo desde o compilador FORTRAN IV para FORTRAN 77, que incluiu processamento vetorial e paralelo e, enfim, para o FORTRAN 8x, finalmente conhecido como FORTRAN 90. Portanto, o FORTRAN 90 é uma extensão do FORTRAN padrão. Ele contém todo o FORTRAN 77, ou seja, todos os programas escritos em FORTRAN 77 ainda são compilados, podendo haver algumas diferenças na forma como serão executados, mas em geral são compatíveis. O FORTRAN 90 foi escrito pensando-se no usuário, do ponto de vista do programador.

3 BIBLIOTECA INTERVALAR BÁSICA

A *libavi.a* é uma biblioteca de rotinas intervalares que implementa a aritmética de alto desempenho, reunindo as características do processamento vetorial com as propriedades da matemática intervalar. A biblioteca de rotinas intervalares *libavi.a* é projetada para viabilizar o uso da matemática intervalar em supercomputadores para a solução de problemas físicos, químicos e das engenharias que necessitem alta exatidão. Para tanto, além da aritmética vetorial intervalar (operações, funções e avaliação de expressões) teve-se que providenciar bibliotecas que tornassem disponíveis a estes usuários os métodos intervalares para solução de seus problemas (ver [DIV95] e [DIV95a]).

A biblioteca *libavi.a* é composta por 290 rotinas intervalares organizadas em quatro módulos. O módulo *básico* inclui o arquivo que contém a definição de intervalos reais e complexos (*inter.inc*). Nesse módulo, são implementadas todas as operações entre intervalos reais. Essas operações servem de base para todos os demais módulos.

O módulo dos intervalos complexos *ci* contém rotinas para manipulação de dados do tipo intervalo complexo. Basicamente, ele contém as mesmas rotinas do módulo *básico* reescritas para esse tipo de dado e também rotinas específicas para intervalos complexos (tais como conjugado). A partir desse módulo estão sendo implementadas rotinas que manipulam vetores e matrizes de intervalos complexos. Para tanto, basta estender as operações complexas para cada elemento do vetor ou matriz de intervalo complexo, de forma análoga à feita com intervalos reais.

Nos módulos *básico* e *ci* existem conjuntos de rotinas ou funções agrupados de acordo com a natureza das operações. Esses conjuntos são seis. O primeiro conjunto é formado pelas funções de transferência, as quais tratam de converter reais ou complexos para intervalos, vice-versa e, ainda, funções que calculam particularidades geométricas dos intervalos, como diâmetro, raio, ponto médio e distância entre intervalos. O segundo conjunto é formado pelas funções relacionais, cujo tipo de dado resultante é geralmente lógico (booleano). Entre as funções relacionais estão a igualdade, a diferença, ser menor, maior, estar contido, conter, ser interior e a relação de pertinência, ou seja, um real pertence ao intervalo. O terceiro conjunto de rotinas trata de operações entre conjuntos. Entre essas operações, estão a interseção, a união e a união convexa.

O quarto conjunto de rotinas implementa a aritmética, ou seja, as operações aritméticas de adição, subtração, multiplicação e divisão. São incluídas algumas operações aritméticas entre diferentes tipos de dados. As funções elementares, como valor absoluto, raiz quadrada, x

elevado ao quadrado, potenciação, exponencial, logaritmo e as trigonométricas constituem o quinto conjunto de rotinas. Por fim, estão as rotinas de entrada e saída, *sread* e *swrite* para os novos tipos de dados.

O módulo *mvi* inclui o módulo *básico* e é incluído no módulo de aplicações *aplic*. Esse módulo implementa todas as operações entre vetores de intervalos, matrizes de intervalos e rotinas de diferentes tipos de dados com vetores e matrizes de intervalos. O módulo de matrizes e vetores intervalares (*mvi*) foi organizado em três partes: a de vetores, a de matrizes de intervalos reais e a parte que contém operações aritméticas de diferentes tipos de dados com intervalos, vetores e matrizes de intervalos reais contendo, cada uma delas, diferentes grupos de rotinas. São eles: Funções de transferência, Operações relacionais e entre conjuntos, Operações aritméticas, Transposição, Funções básicas e Rotinas de entrada e saída.

O último módulo é o das aplicações, denominado *aplic*. Nesta parte existem algumas rotinas que implementam operações aritméticas compostas existentes em outras bibliotecas. Para a escolha dessas rotinas, foram analisadas algumas bibliotecas, como a *NUMERALS*, da Bourroghs, as BLAS e o Pascal-XSC. O módulo *aplic* inclui o módulo *mvi* e o *básico*.

3.1 Núcleo de Aritmética de Alta Exatidão

As quatro operações básicas (adição, subtração, multiplicação e divisão) em ponto-flutuante são realizadas em unidades funcionais especiais. São três unidades funcionais de ponto-flutuante Elas realizam a aritmética para operações entre escalares e vetores. A unidade funcional de adição realiza a operação de soma e subtração de operandos. A unidade funcional de multiplicação realiza a operação de multiplicação. O Cray não possui uma unidade funcional dedicada à operação de divisão. Ela é realizada em dois passos, por exemplo, achar o quociente de A/B. Primeiro, é calculado o recíproco de B, 1/B, na unidade funcional de aproximação recíproca. Então, esse resultado é multiplicado por A. A mantissa possui 48 bits, em precisão simples e 96 em precisão dupla. O tipo de arredondamento disponível no Cray é o arredondamento para o número mais próximo.

A biblioteca *libavi.a* foi desenvolvida com a finalidade de explorar o alto desempenho do Cray no cálculo com a aritmética intervalar. Os resultados são confiáveis, pois estão sempre contidos nos intervalos resultantes. Mas vale ressaltar que os números em ponto-flutuante não são representados de acordo com o Padrão de aritmética binária (IEEE 754). E que os arredondamentos direcionados foram simulados através de funções do Fortran 90, que calculam o antecessor e o sucessor de um número. Esta implementação não é exatamente a definição de arredondamentos direcionados. Ela introduz algum erro, já que o intervalo é inflacionado mais do que o necessário. Sendo assim, a implementação não é com máxima exatidão.

No supercomputador Cray, portanto, como não está disponível essa aritmética de alta exatidão, muitos cálculos produzem resultados de má qualidade, tanto no modo seqüencial como no modo vetorial. Estudos anteriores ([FER95] e [DIV96]) comprovaram que existem falhas nos cálculos quando a ordem das operações são trocadas e quando é usado o processamento vetorial. Portanto, foi constatado que no Cray, resultados diferentes são obtidos para uma mesma operação quando são executados nos modos de processamento seqüencial e vetorial. As causas disto são, entre outras, a inexistência de uma aritmética vetorial de alta exatidão ([IMA91]) e a sensibilidade dos cálculos à troca de ordem (instabilidade).

Para a obtenção da mesma, há uma grande lacuna a ser preenchida. Para isto, está sendo implementado um núcleo de aritmética de alta exatidão, que será posteriormente incorporado à *libavi.a* (desenvolvida em Fortran 90), incluindo os arredondamentos direcionados, as quatro operações com máxima exatidão e o produto escalar ótimo. O Núcleo é o módulo mais básico do ambiente da biblioteca *libavi.a*. Será implementado inteiramente por *software*.

4 MÓDULOS APLICATIVOS INTERVALARES

Um dos objetivos do Laboratório de Aritmética de Alta Exatidão e de Alto Desempenho é a elaboração de ferramentas computacionais que utilizam a aritmética intervalar na resolução de problemas da computação científica e da computação nas mais diversas áreas de pesquisa, como por exemplo problemas que envolvam a resolução de sistemas de equações lineares (ver [HÖL96]).

O principal objetivo do desenvolvimento da biblioteca aplicativa intervalar para a resolução de sistemas de equações lineares (*libselint.a*) é a utilização dessa biblioteca na difusão do estudo de métodos intervalares junto aos meios acadêmicos e a pessoas que estejam interessadas a respeito desses métodos. A criação de bibliotecas científicas independentes, justifica-se no fato de que estas devem ser incluídas nos programas aplicativos dos usuários.

A *libselint.a* é uma biblioteca desenvolvida em Fortran 90 e seu nome significa: *lib* - biblioteca em inglês, *int* - intervalos, *.a* - sufixo padrão de biblioteca no ambiente UNIX. Ela é o primeiro módulo aplicativo implementado do Laboratório de Alta Exatidão e de Alto Desempenho. Essa biblioteca é composta de 24 rotinas que implementam 18 métodos para a solução de sistemas de equações lineares intervalares e pontuais (ver [HÖL96] e [HAM93]). As padronizações e convenções de especificação e documentação seguem o padrão da *Cray Research Inc*. A *libselint.a* é organizada em quatro módulos:

- *dirint*: inclui os métodos baseados em operações algébricas intervalares e propriedades intervalares, que são também conhecidos como métodos diretos intervalares. Seis métodos foram implementados, sendo os Métodos de Hansen os mais conhecidos deles;

- *refint*: inclui os métodos baseados em inclusões ou refinamentos intervalares da solução e do erro. Pode-se fazer uso de uma solução inicial calculada através de métodos pontuais. Então, ela é refinada através de métodos intervalares. Nos métodos disponíveis, o refinamento pode ocorrer na solução ou no resíduo, através do uso de matrizes inversas intervalares ou pontuais;

- *itrint*: inclui os métodos iterativos intervalares. Eles também são conhecidos como métodos de relaxação e são baseados em inclusões monotônicas. Estes métodos são usados em grandes sistemas esparsos;

- *equalg*: contém as rotinas para a resolução de sistemas lineares de ordem um, isto é, solução algébrica de equações através de métodos intervalares. Os métodos

disponíveis foram principalmente baseados nas versões intervalares do Método de Newton, isto é, diferentes maneiras de calcular o operador Newtoniano.

Para o uso da biblioteca *libselint.a*, também deve ser incluído no programa aplicativo do usuário o arquivo *inter.inc*, que contém as definições de intervalos reais e complexos, e a biblioteca *libavi.a*, uma vez que as rotinas necessárias de manipulação de intervalos, vetores e matrizes de intervalos foram ali implementadas. Pode ser necessária ainda a inclusão da biblioteca *libsci.a*, pois algumas das rotinas do módulo aplic se utilizam de rotinas da biblioteca BLAS.

4.1 Características e Documentação

A *libavi.a* é uma biblioteca de rotinas básicas de intervalos e a *libselint.a* é uma biblioteca aplicativa intervalar, ambas para o ambiente do supercomputador Cray. As padronizações e convenções de especificação e documentação seguem o padrão da *Cray Research Inc*.

Como o propósito do desenvolvimento destas bibliotecas foi o de providenciar ferramentas úteis para a programação científica (em ambiente vetorial), as rotinas implementadas necessitavam de certas características, como exatidão, eficiência, confiabilidade, validação, facilidade de uso, boa documentação, modificabilidade, completude e transportabilidade.

A exatidão dessas rotinas foi obtida pelo uso das rotinas disponíveis no Fortran 90 para manipulação de números reais (números em ponto-flutuante), especialmente as funções básicas elementares. Para obter melhor exatidão, utilizaram-se algumas rotinas em dupla precisão, especialmente para se tentar ter um produto escalar mais próximo do ótimo. Utilizaram-se, ainda, algumas das rotinas da biblioteca BLAS para a implementação das operações entre vetores e matrizes de intervalos. A extensão para rotinas intervalares se deu através da aplicação de princípios de conversão real-intervalo e das propriedades de semimorfismo. Manteve-se a exatidão disponível do supercomputador Cray para processamento científico, mas se acrescentou, com intervalos, a garantia dos resultados.

Em relação à eficiência, evoluiu-se mais neste conceito, pois inicialmente eficiência significava um uso otimizado de memória, por ser muito cara e escassa. Com a evolução tecnológica, eficiência se vinculou à idéia de velocidade de processamento. Perdeu-se um pouco da idéia de qualidade para se ter velocidade. Com a matemática intervalar (acrescida de aritmética de alta exatidão) é resgatado o conceito de eficiência como qualidade e garantia do resultado calculado com rapidez. A biblioteca *libavi.a* produz resultados um pouco mais lentos do que a aritmética de ponto-flutuante ordinária, uma vez que os cálculos são efetuados para os extremos dos intervalos, mas ganha-se na garantia dos resultados.

As qualidades de confiabilidade e validação dizem respeito ao fato de a biblioteca ser consistente com a documentação, ou seja, a execução do programa realiza exatamente o que seu propósito descrito na documentação define, resolvendo a classe de problemas a que se propõe. A documentação da biblioteca de rotinas intervalares é constituída do manual de utilização da *libavi.a* e do manual de utilização da *libselint.a*. Eles são constituídos por duas partes, a primeira contém informações sobre o uso da biblioteca e a outra parte contém a

documentação das rotinas. A primeira parte visa atender um usuário que deseja maiores informações da biblioteca, enquanto que a segunda parte visa atender ao uso esporádico.

5 CONCLUSÕES

Este artigo apresentou a aritmética de alta exatidão, onde são descritas as suas principais características. E, ainda, a importância de se ter qualidade nas operações em ponto-flutuante, pois em computações de larga escala os erros se acumulam, podendo gerar resultados errados.

Com o desenvolvimento deste trabalho se conseguiu uma confiabilidade nos cálculos realizados no supercomputador CRAY Y-MP, não só pela maior qualidade obtida através do uso do Núcleo de Aritmética de Alto desempenho, mas também porque a solução dos problemas que são resolvidos é dada através de intervalos que possuem limites confiáveis. Através deste ambiente, a matemática intervalar foi levada para o campo da prática e tornou-se possível o seu uso na resolução de problemas numéricos, inclusive no caso de problemas instáveis, nos quais a solução gerada era totalmente incorreta. Por fim, as bibliotecas *libavi.a* e *libselint.a* tornam-se ferramentas úteis na resolução de problemas práticos de grande porte, estando disponíveis aos usuários no Centro Nacional de Supercomputação (CESUP/UFRGS).

6 REFERÊNCIAS BIBLIOGRÁFICAS

[CRA93] CRAY RESEARCH, Inc. CF90 Fortran language reference manual., n.sr.3902, versão 1.0. [S.l : s.n.], 1993.

[DIV95] DIVERIO, T. A. Uso efetivo da matemática intervalar em supercomputadores vetoriais. Porto Alegre: PGCC da UFRGS, 1995. Tese de doutorado.

[DIV95a] DIVERIO, T. A. LIBAVI.A Manual de utilização. Porto Alegre: PGCC da UFRGS, 1995. 350p.

[DIV96] DIVERIO, T. A.; FERNANDES, U. A. L.; CLAUDIO, D.M. Error processing and the library libavi.a. **Reliable Computing**, v.2 n.2 p.103-110, 1996.

[FER95] FERNANDES, U. A. L.; DAHMER,A.; DIVERIO, T. A. Limitações do Processamento vetorial no Cray Y-MP2E. Porto Alegre: PGCC da UFRGS, 1995. 60p. (RP 248).

[HAM93] HAMMER, R. et al. Numerical Toolbox for Verified Computing I: basic numerical problems. Berlim: Springer-Verlag, 1993. 337p.

[HOL96] HÖLBIG, C.A. Métodos Intervalares para a Resolução de Sistemas de Equações Lineares. Porto Alegre: PGCC da UFRGS, 1996. Dissertação de mestrado.

[IMA91] IMACS, GAMM. Resolution on computer arithmetic. In: Computer Arithmetic, Scientific Computation and Mathematical Modelling. KAUCHER, E.; MARKOV, S. M.; MAYER, G. (Eds.). Basel: J.C.Baltzer, 1991. (Proceedings of SCAN'90, IMACS Annals on Computing and Applied Mathematics, Albena, Sept 24-28, 1990). v.12. p.477-479.

[KUL81] KULISCH,U.W; MIRANKER,W.L Computer arithmetic in theory and practice, 1981.

[KUL83] KULISCH, U. W.; MIRANKER, W. L. A new approach to scientific computation. New York: Academic Press, 1983. 384p.